Machine Learning

AIML CZG565

M6 : Instance Based Learning

Session 9

22 March 26

BITS Pilani

Pilani Campus

Machine Learning

innovate achieve lead

Disclaimer and Acknowledgement

•These content of modules & context under topics are planned by the course owner Dr. Sugata, with grateful acknowledgement to many others who made their course materials freely available online

•We here by acknowledge all the contributors for their material and inputs.

•We have provided source information wherever necessary

•Students are requested to refer to the textbook w.r.t detailed content of the presentation deck shared over canvas

•We have reduced the slides from canvas and modified the content flow to suit the requirements of the course and for ease of class presentation

Slide Source / Preparation / Review:
From BITS Pilani WILP**:** Prof.Sugata, Prof.Chetana, , Prof.Monali, Prof.Rajavadhana, Prof.Seetha, Prof.Anita, Prof. Swarna

External: CS109 and CS229 Stanford lecture notes, Dr.Andrew NG and many others who made their course materials freely available online

Course Plan

M1	Introduction
M2	Machine learning Workflow
M3	Linear Models for Regression
M4	Linear Models for Classification
M5	Decision Tree
M6	Instance Based Learning
M7	Support Vector Machine
M8	Bayesian Learning
M9	Ensemble Learning
M10	Unsupervised Learning
M11	Machine Learning Model Evaluation/Comparison

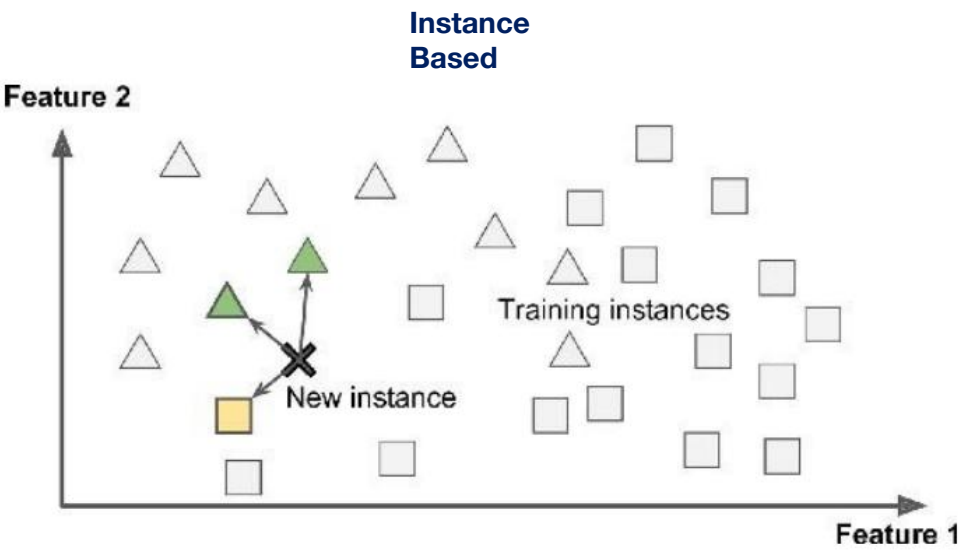
Instance Based Learning

Introduction to Machine Learning

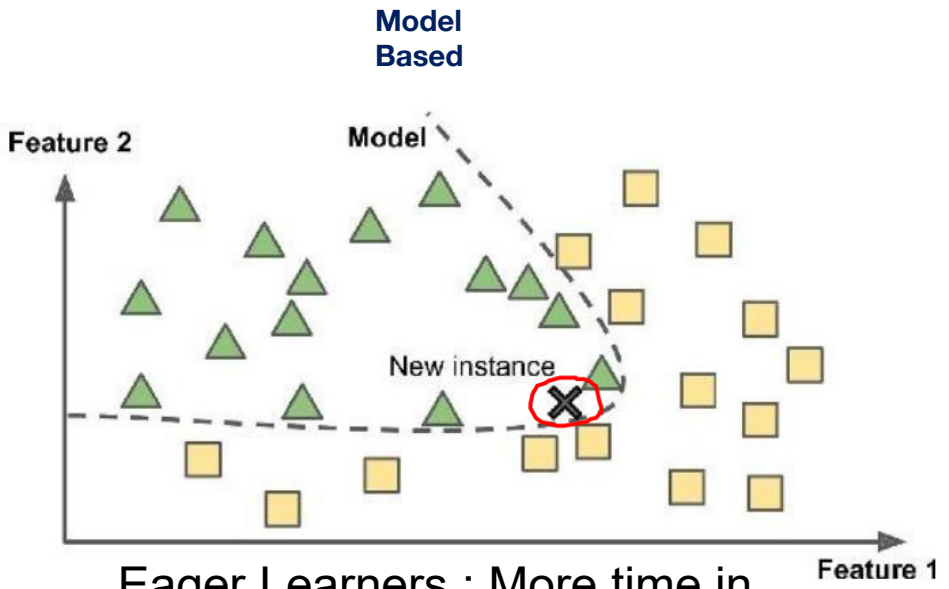
innovate achieve lead

Types: Based on how & when the training data is used

- Instance Based Learning: Compare new data points to known data points
- Model Based learning : Detect patterns in the training data and build a predictive model



Lazy Learners : Less time in training but more time in predicting



Eager Learners : More time in training but Less time in predicting

Instance Based Learning

Model-based learning techniques

Use the input data

$$\begin{bmatrix} x_{1,0} & x_{1,1} & \dots & x_{1,n} \\ x_{2,0} & x_{2,1} & \dots & x_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{m,0} & x_{m,1} & \cdots & x_{m,n} \end{bmatrix} \text{ and } \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_m \end{bmatrix}$$

m x n

To learn a set of parameters

$$\begin{bmatrix} \theta_0 & \theta_1 & \cdots & \theta_n \end{bmatrix}$$

Which yield a **generalized** function

$$f(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$

Capable of predicting values or classes on new input data

$$f(x_i) = 39$$

$$f(x_j) = 1$$

Instance-based learning techniques

Store the input data

$$\begin{bmatrix} x_{1,0} & x_{1,1} & \dots & x_{1,n} \\ x_{2,0} & x_{2,1} & \dots & x_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{m,0} & x_{m,1} & \cdots & x_{m,n} \end{bmatrix} \text{ and } \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_m \end{bmatrix}$$

When asked to predict a new value (a query)

$$y_i = ?$$

Search for similar data points previously stored

$$\begin{bmatrix} x_{4,1} & x_{4,2} & \cdots & x_{4,n} \\ x_{9,1} & x_{9,1} & \cdots & x_{9,n} \\ x_{15,1} & x_{15,1} & \cdots & x_{15,n} \end{bmatrix} \text{ and } \begin{bmatrix} y_4 \\ y_9 \\ y_{15} \end{bmatrix}$$

And use them to generate your prediction

$$y_i = \frac{y_4 + y_9 + y_{15}}{3}$$

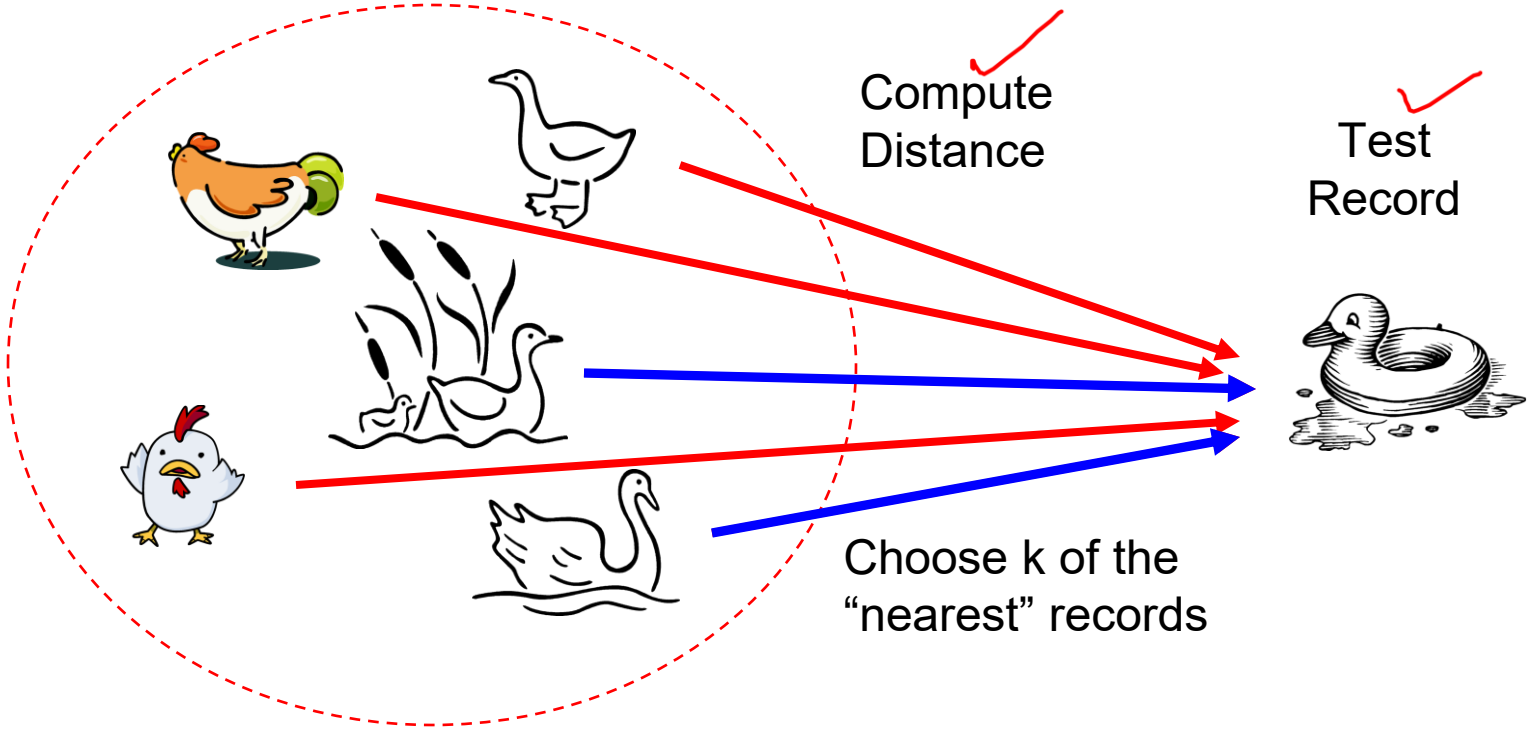
Source Credit : https://www.jeremyjordan.me/k-nearest-neighbors/

Instance Based Learning

Nearest Neighbor Approach

Basic idea:

- If it walks like a duck, quacks like a duck, then it's probably a duck



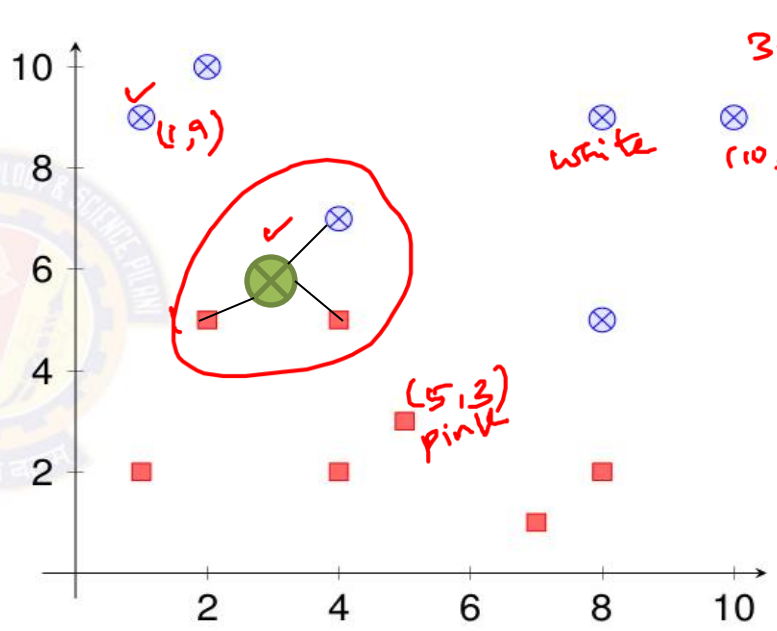
Instance Based Learning

Nearest Neighbor Approach : Notations

K-nearest neighbor:

- Given x_q, take **vote** among its k nearest neighbors (if **discrete-valued target function**)
- Take **mean(or median)** of f values of k nearest neighbors (if **real-valued**) $f^*(x_q)=\Sigma_{i=1}^{k} f(x_i)/k$

x_1	x_2	y
1	9	white
10	9	white
4	7	white
4	5	pink
5	3	pink
8	9	white
4	2	pink
2	5	pink
7	1	pink
2	10	white
8	5	white
1	2	pink
8	2	pink



Interpretation : Assume that the Instances represented as points in a Euclidean space

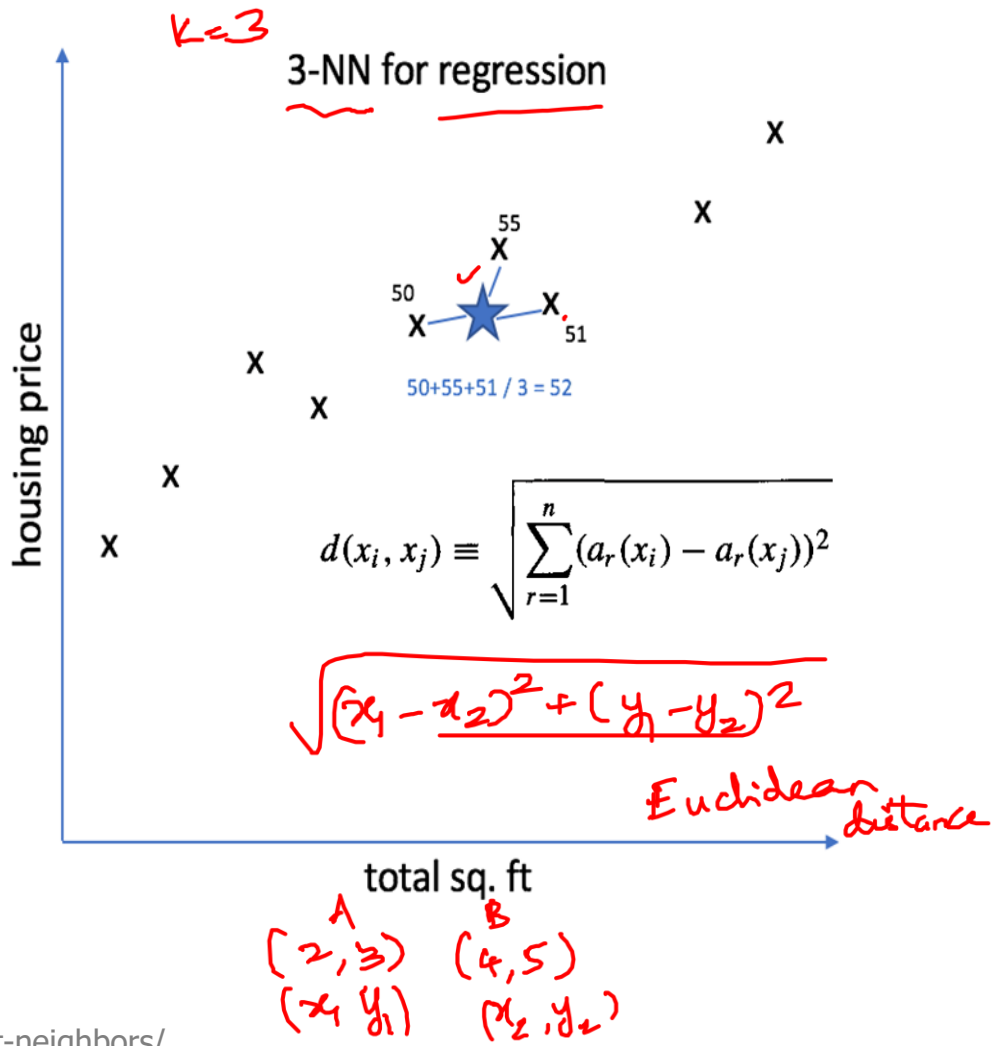
Instance Based Learning

K-Nearest Neighbor Regression: Example

Size in feet² (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
1852	178
…	…
…	??…

real values

Mean Euclidean distance



Source Credit : https://www.jeremyjordan.me/k-nearest-neighbors/

Measures of Dis/similarity:

Data Matrix and Distance or dissimilarity Matrix

Data matrix

- n data points with p dimensions

$$\begin{bmatrix} x_{11} & \cdots & x_{1f} & \cdots & x_{1p} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{i1} & \cdots & x_{if} & \cdots & x_{ip} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{n1} & \cdots & x_{nf} & \cdots & x_{np} \end{bmatrix}$$

n x p

Distance / dissimilarity matrix

- n data points, but registers only the distance
- A triangular matrix

$$\begin{array}{c} \\ A \\ B \\ C \\ \vdots \\ Z \end{array} \begin{array}{c} \begin{array}{ccccc} A & B & C & D & \cdots \end{array} \\ \begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ : & : & : & & \\ d(n,1) & d(n,2) & \cdots & \cdots & 0 \end{bmatrix} \end{array}$$

$d(A,B) = d(B,A)$

$d(i,i) = 0$

$d(i,j) = d(A,B) = d(2,1) = d(1,2)$

Measures of Distance/similarity:

Nominal Attributes

Can take 2 or more states, e.g., red, yellow, blue, green (generalization of a binary attribute)

Method 1: Simple matching

– m: # of matches, p: total # of variables $d(i,j) = \frac{p-m}{p}$

How it works: This method treats all M states of the nominal attribute equally. It focuses simply on whether the states for a given attribute are the same or different between the two objects being compared.

A "match" occurs when the two objects have the exact same state for a specific attribute (e.g., both are "red").

This method is best when there is **no meaningful order or hierarchy** among the nominal states.

Example: If you're comparing two cars based on three attributes (Color, Make, Model) and they match on Color (both "Blue") and Make (both "Honda") but not Model, then m=2 and p=3. The similarity is 2/3.

distance = 1 − similarity = 1/3

Method 2: Use a large number of binary attributes

- creating a new binary attribute for each of the *M* nominal states

This method, often called **one-hot encoding** or **dummy coding**, transforms the single M-state nominal attribute into M new, independent binary attributes.

How it works:
For an attribute with M nominal states (e.g., Color: {Red, Yellow, Blue}, so M=3), you **create M new binary attributes**.
In the example above, you'd create three new attributes: ColorRed, ColorYellow, and ColorBlue.
For any given object, **only one** of these M new binary attributes will be set to '1' (or 'True'), and the rest will be '0' (or 'False').

Example:
An object with Color="Red" would be represented by:
{ColorRed=1,ColorYellow=0,ColorBlue=0}

An object with Color="Blue" would be represented by:
{ColorRed=0,ColorYellow=0,ColorBlue=1}

Benefit:

Once the nominal attribute is transformed into a set of binary attributes, you can use standard distance or similarity measures designed for binary data, such as the **Jaccard coefficient** or the **Simple Matching Coefficient** (as discussed in Method 1), on the new set of binary variables. This allows the nominal attribute to be seamlessly integrated with other types of attributes (like numeric or true binary) in complex analysis models.

While effective, this method **increases the dimensionality** of the dataset which can sometimes complicate analysis, especially if M is very large.

Measures of Distance/similarity: Binary Attributes

- A contingency table for binary data

Object *i* \ Object *j*	1	0	sum
1	q	r	$q+r$
0	s	t	$s+t$
sum	$q+s$	$r+t$	p

- Distance measure for symmetric binary variables:

$$d(i,j) = \frac{r+s}{q+r+s+t}$$

- Distance measure for asymmetric binary variables:

$$d(i,j) = \frac{r+s}{q+r+s}$$

- Jaccard coefficient (*similarity* measure for *asymmetric* binary variables):

$$sim_{Jaccard}(i,j) = \frac{q}{q+r+s}$$

Always choose the right measure depending on **what kind of binary variable** you're working with.

Measures of Dis/similarity: Binary Attributes

Example

Name	Gender	Fever	Cough	Test-1	Test-2	Test-3	Test-4
Jack	M	Y	N	P	N	N	N
Mary	F	Y	N	P	N	P	N
Jim	M	Y	P	N	N	N	N

- Gender is a symmetric attribute
- The remaining attributes are asymmetric binary
- Let the values Y and P be 1, and the value N is 0

$$d(jack, mary) = \frac{0+1}{2+0+1} = 0.33$$

$$d(jack, jim) = \frac{1+1}{1+1+1} = 0.67$$

$$d(jim, mary) = \frac{1+2}{1+1+2} = 0.75$$

Under the Jaccard/Asymmetric Binary method, **Mary and Jim** are the **most dissimilar** (highest distance), and **Jack and Mary** are the **most similar** (lowest distance).

Binary Mapping:

- 'Y' or 'P' → **1** (Significant/Present)
- 'N' or 'F' → **0** (Non-significant/Absent)

d(jack, jim) =

Name	Fever (1/0)	Cough (1/0)	Test-1 (1/0)	Test-2 (1/0)	Test-3 (1/0)	Test-4 (1/0)
Jack	1	0	1	0	0	0
Mary	1	0	1	0	1	0
Jim	1	1	0	0	0	0

The variables used are p=6 (Fever, Cough, Test-1, Test-2, Test-3, Test-4).

Let's define the components for two objects i and j:

- a: Number of attributes where i=1 and j=1 (Significant Matches)
- b: Number of attributes where i=1 and j=0 (Jack/Mary/Jim is '1', the other is '0')
- c: Number of attributes where i=0 and j=1 (Jack/Mary/Jim is '0', the other is '1')
- d: Number of attributes where i=0 and j=0 (Insignificant Matches - **Ignored**)

Comparing Jack (i) and Mary (j)

Attribute	Jack	Mary	Result
Fever	1	1	a
Cough	0	0	d (Ignored)
Test-1	1	1	a
Test-2	0	0	d (Ignored)
Test-3	0	1	c
Test-4	0	0	d (Ignored)

$a = 2$ (Fever, Test-1)

$b = 0$

$c = 1$ (Test-3)

$a + b + c = 2 + 0 + 1 = 3$

Jaccard Similarity Coefficient:

$$\text{Sim}(i, j) = J(i, j) = \frac{a}{a + b + c}$$

Jaccard Distance:

$$\text{Dist}(i, j) = 1 - J(i, j) = \frac{b + c}{a + b + c}$$

$$\text{Dist}(\text{Jack}, \text{Mary}) = \frac{b + c}{a + b + c} = \frac{0 + 1}{3} = \frac{1}{3} \approx \mathbf{0.333}$$

Comparing Jack (i) and Jim (j)

Attribute	Jack	Jim	Result
Fever	1	1	a
Cough	0	1	c
Test-1	1	0	b
Test-2	0	0	d (Ignored)
Test-3	0	0	d (Ignored)
Test-4	0	0	d (Ignored)

$a = 1$ (Fever)

$b = 1$ (Test-1)

$c = 1$ (Cough)

$a + b + c = 1 + 1 + 1 = 3$

$$\text{Dist}(\text{Jack}, \text{Jim}) = \frac{b + c}{a + b + c} = \frac{1 + 1}{3} = \frac{2}{3} \approx \mathbf{0.667}$$

Comparing Mary (i) and Jim (j)

Attribute	Mary	Jim	Result
Fever	1	1	a
Cough	0	1	c
Test-1	1	0	b
Test-2	0	0	d (Ignored)
Test-3	1	0	b
Test-4	0	0	d (Ignored)

$a = 1$ (Fever)

$b = 2$ (Test-1, Test-3)

$c = 1$ (Cough)

$a + b + c = 1 + 2 + 1 = 4$

$$\text{Dist}(\text{Mary}, \text{Jim}) = \frac{b + c}{a + b + c} = \frac{2 + 1}{4} = \frac{3}{4} = \mathbf{0.750}$$

Example:

Market Basket Analysis

Customer	Milk	Bread	Eggs	Butter
A	1	0	1	1
B	1	1	0	1

Compare A and B:

q = 2 (Milk & Butter)

r = 1 (Eggs only A)

s = 1 (Bread only B)

t = 0 (No shared absences — we ignore)

Jaccard similarity = 2 / (2 + 1 + 1) = 0.5

Jaccard distance = 0.5

Measures of Dis/similarity

Distances: Minkowski Distance

Minkowski distance: A popular distance measure

$$d(i, j) = \sqrt[h]{|x_{i1} - x_{j1}|^h + |x_{i2} - x_{j2}|^h + \cdots + |x_{ip} - x_{jp}|^h}$$

where $i = (x_{i1}, x_{i2}, \ldots, x_{ip})$ and $j = (x_{j1}, x_{j2}, \ldots, x_{jp})$ are two p-dimensional data objects, and h is the order (the distance so defined is also called L-h norm)

Properties

- $d(i, j) > 0$ if $i \neq j$, and $d(i, i) = 0$ (Positive definiteness)
- $d(i, j) = d(j, i)$ (Symmetry)
- $d(i, j) \leq d(i, k) + d(k, j)$ (Triangle Inequality)

A distance that satisfies these properties is a metric

Measures of Dis/similarity

Distances : Special Cases of Minkowski

- $h = 1$: Manhattan (city block, L_1 norm) distance
  - E.g., the Hamming distance: the number of bits that are different between two binary vectors

$$d(i,j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \ldots + |x_{ip} - x_{jp}|$$

- $h = 2$: (L_2 norm) Euclidean distance

$$d(i,j) = \sqrt{(|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \ldots + |x_{ip} - x_{jp}|^2)}$$

- $h \to \infty$. "supremum" (L_{max} norm, L_∞ norm) distance.
  - This is the maximum difl

$$d(i,j) = \lim_{h \to \infty} \left(\sum_{f=1}^{p} |x_{if} - x_{jf}|^h \right)^{\frac{1}{h}} = \max_{f}^{p} |x_{if} - x_{jf}|$$

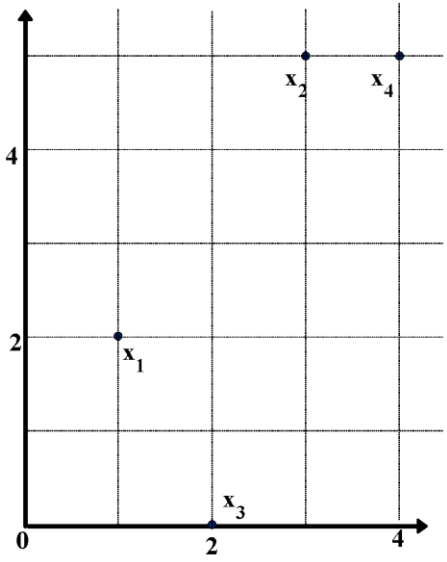
Example: Minkowski Distance

$x_1 = (1,2)$
$x_2 (3,5)$

(Dissimilarity Matrices)

$d(x_2, x_4) = |3-1| + |5-2| = 2+3 = 5$

point	attribute 1	attribute 2
x1	1	2
x2	3	5
x3	2	0
x4	4	5



Manhattan (L_1)

L	x1	x2	x3	x4
x1	0			
x2	5	0		
x3	3	6	0	
x4	6	1	7	0

Euclidean (L_2)

L2	x1	x2	x3	x4
x1	0			
x2	3.61	0		
x3	2.24	5.1	0	
x4	4.24	1	5.39	0

$d(x_2, x_4) = \sqrt{(3-1)^2 + (5-2)^2} = \sqrt{2^2 + 3^2} = \sqrt{13}$

Supremum

L_∞	x1	x2	x3	x4
x1	0			
x2	3	0		
x3	2	5	0	
x4	3	1	5	0

Try

$x_4 = (1,2)$ $x_3 = (2,0)$

Manhattan distance:

$$d(x_3, x_4) = |2-1| + |2-0| = 1+2 = 3$$

Euclidean distance:

$$d(x_3, x_4) = \sqrt{(2-1)^2 + (2-0)^2} = \sqrt{1^2+2^2} = \sqrt{5}$$

Standardizing Numeric Data

- Z-score:
  - X: raw score to be standardized, μ: mean of the population, σ: standard deviation
  - the distance between the raw score and the population mean in units of the standard deviation
  - negative when the raw score is below the mean, "+" when above

Where

$$z = \frac{x - \mu}{\sigma}$$

Normal distribution

$$Z = \frac{\bar{x} - \mu}{\sigma}$$

$Z \rightarrow -ve \quad x < \mu$

$Z \rightarrow +ve \quad x > \mu$

Measures of Dis/similarity : Ordinal Variables

- An ordinal variable can be discrete or continuous
- Order is important, e.g., rank or grades
- Can be treated like interval-scaled
  - replace x_{if} by their rank
  - map the range of each variable onto [0, 1] by replacing i-th object in the f-th variable by

$$r_{if} \in \{1, \ldots, M_f\}$$

no. of distinct ranks

  - compute the dissimilarity using methods for interval-scaled variables

$$z_{if} = \frac{r_{if} - 1}{M_f - 1}$$

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

Eg: Suppose variable f is like a rating with 4 levels:

1. Poor
2. Fair
3. Good
4. Excellent

So, $N_f = 4$

Step 1: Assign ranks

Person	Satisfaction	rank r_{if}
A	Poor	1
B	Good	3
C	Excellent	4

Step 2: Normalization

$$z_{if} = \frac{r_{if} - 1}{M_f - 1}$$ ✓

d(Ram, Kishan)

Z(A, B)

$$\frac{r(A) - r(B)}{M_f - 1}$$

Person A : $z = \frac{1-1}{4-1} = 0$

Person B : $z = \frac{3-1}{4-1} = \frac{2}{3} \approx 0.67$

Person C : $z = \frac{4-1}{4-1} = 1$

This lets us to treat ordinal data like a number

Measures of Dis/similarity
Attributes of Mixed Type: Gower distance

- A database may contain all attribute types
  - Nominal, symmetric binary, asymmetric binary, numeric, ordinal

$$d(i,j) = \frac{\sum_{c=1}^{n} \omega_c \delta_{ij}^{(c)} d_{ij}^{(c)}}{\sum_{c=1}^{n} \omega_c \delta_{ij}^{(c)}}$$

$d(i,j)$ = dissimilarity between row i and row j

c = the cth column

n = number of columns in the dataset

ω_c = weight of cth column $= \frac{1}{\text{nrows in dataset}}$

$$\delta_{ij}^{c} = \begin{cases} 0 & \text{if column c is missing in row i or j} \\ 0 & \text{if column c is asymmetric binary and both values in row i and j are 0} \\ 1 & \text{otherwise} \end{cases}$$

$$d_{ij}^{c}(\text{categorical}) = \begin{cases} 0 & \text{if i and j are equal in column c} \\ 1 & \text{otherwise} \end{cases}$$

$$d_{ij}^{c}(\text{continuous/ordinal}) = \frac{|\text{row i in column c - row j in column c}|}{\max(\text{column c}) - \min(\text{column c})}$$

https://healthcare.ai/clustering-non-continuous-variables/

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

Example

Based on the information given in the table below, find most similar and most dissimilar persons among them. Apply min-max normalization on income to obtain [0,1] range. Consider profession and mother tongue as nominal. Consider native place as ordinal variable with ranking order of [Village, Small Town, Suburban, Metropolitan]. Give equal weight to each attribute.

Name	Income	Profession	Mother tongue	Native Place
Ram	70000	Doctor	Bengali	Village
Balram	50000	Data Scientist	Hindi	Small Town
Bharat	60000	Carpenter	Hindi	Suburban
Kishan	80000	Doctor	Bhojpuri	Metropolitan

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

Normalization:

$$v' = \frac{v - \min_A}{\max_A - \min_A}$$

The normalized values (v′) for each income are calculated as follows:

Original Income (v)	Calculation	Normalized Income (v')
70000	$\frac{70000-50000}{30000} = \frac{20000}{30000}$	$\approx **0.6667**$
50000	$\frac{50000-50000}{30000} = \frac{0}{30000}$	**0.0000**
60000	$\frac{60000-50000}{30000} = \frac{10000}{30000}$	$\approx **0.3333**$
80000	$\frac{80000-50000}{30000} = \frac{30000}{30000}$	**1.0000**

Solution

After normalizing income and quantifying native place, we get

Name	Income	Profession	Mother tongue	Native Place
Ram	0.67	Doctor	Bengali	1
Balram	0	Data Scientist	Hindi	2
Bharat	0.33	Carpenter	Hindi	3
Kishan	1	Doctor	Bhojpuri	4

Normalization $\frac{2-1}{4-1} = \frac{1}{3} = 0.33$

d(Ram, Balram) = 0.67+1+1+(2-1)/(4-1)=3 d(Ram, Bharat) = 0.33+1+1+(3-1)/(4-1)=3

d(Ram, Kishan) = 0.33+0+1+(4-1)/(4-1) = 2.33 d(Balram, Bharat) = 0.33+1+0+(3-2)/(4-1)=1.67

d(Balram, Kishan) = 1+1+1+(4-2)/(4-1) = 3.67 d(Bharat, Kishan) = 0.67+1+1+(4-3)/(4-1) = 3

Most similar – Balram and Bharat; Most dissimilar – Balram and Kishan

Use $kd(i,j) = \frac{1}{d(i,j)^2}$

& find d(Ram, Kishan)

Summary:

Distance between two individuals:

$$d = \underbrace{|\ Income_1 - Income_2\ |}_{Normalized} + \underbrace{MT\ Diff}_{0\ or\ 1} + \underbrace{EduDiff}_{0\ or\ 1} + \underbrace{\frac{|\ NP_1 - NP_2\ |}{3}}_{Normalized\ Native\ Place}$$

Where:

Income is **normalized to [0, 1]**

Profession and Mother tongue are **binary**

Native Place is **quantified nominal**, normalized by dividing by max difference (4 - 1 = 3)

Example:

Assume:

Ram income = 0.67, Kishan income = 1 → |0.67 - 1| = 0.33

Profession: Ram (Doctor), Kishan (Doctor) → Same → 0

Mother tongue: Ram (Bengali), Kishan (Bhojpuri) → Different → 1

Native Place: Ram = 1, Kishan = 4 → |4 - 1| = 3 → 3 / (4 - 1) = 1

Add it up:

$$0.33 + 0 + 1 + 1 = 2.33$$

Gower distance $\frac{\gamma_{if} - 1}{M_f - 1}$

d(Ram, Kishan) = 0.33 + 0 + 1 + (4 - 1)/(4 - 1) = 2.33

d(Kishan, Ram) = 2.33

Example:

Assume:

Balram income = 0, Bharat income = 0.33 → |0 – 0.33| = 0.33

Profession : Balram (Data scientist), Bharat (Carpenter) → Different → 1

Mother tongue : Hindi → Same → 0

Native place: Balram = 2, Bharat = 3 → |3 - 2| = 1 → 1 / (4 - 1) = 1/3 ≈ 0.33

So total:

$$0.33 + 1 + 0 + 0.33 = 1.66 \approx 1.67$$

d(Balram, Bharat) = 0.33 + 1 + 0 + (3 - 2)/(4 - 1) = 1.67

Measures of Dis/similarity

Curse of Dimensionality

•It uses all attributes to classify instances

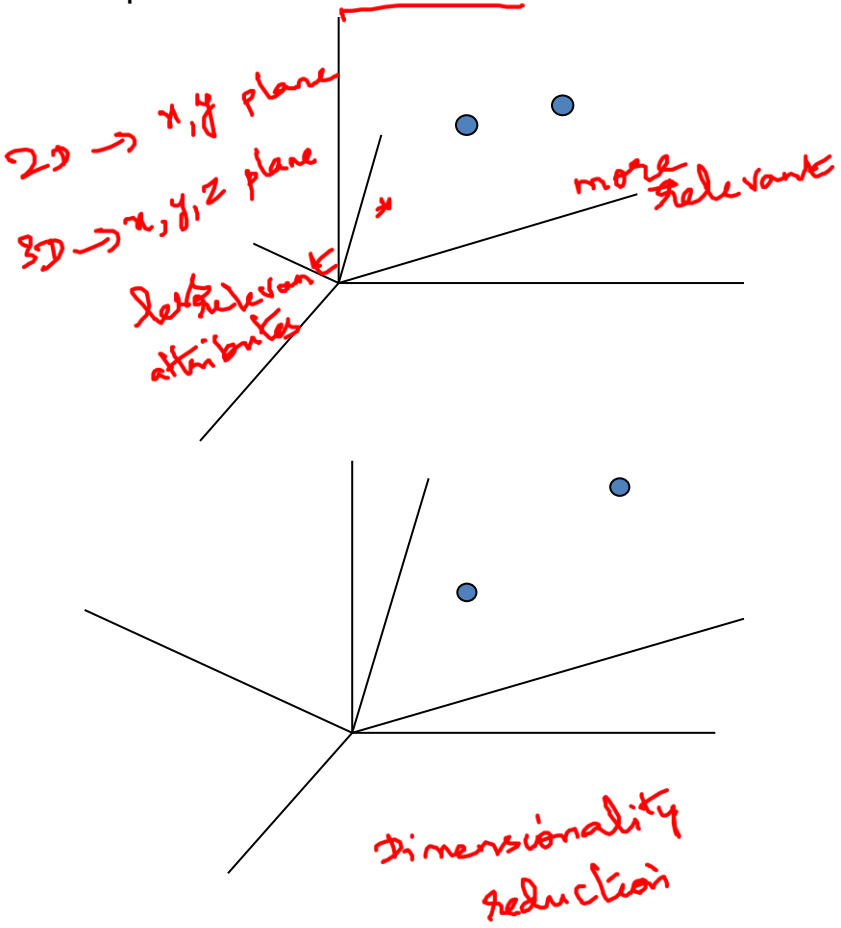
•If the target concept depends on only a few of the many available attributes, then the instances that are truly most "similar" may well be a large distance apart.

•***Curse of dimensionality***: nearest neighbor is easily misled when instance space is high-dimensional

Solution 1: weigh the attributes differently (use cross-validation to determine the weights)

Solution 2: eliminate the least relevant attributes (again, use cross-validation to determine which attributes to eliminate)

Stretching the axes in the Euclidean space, shortening the axes that correspond to less relevant attributes, and lengthening the axes that correspond to more relevant attributes



Instance Based Learning

innovate achieve lead

K-Nearest Neighbor : Algorithm

Approximating a discrete-valued function $f : \Re^n \rightarrow V$ → set of all possible class labels

Training algorithm:

- For each training example $\langle x, f(x) \rangle$, add the example to the list *training_examples*

input - output pairs

Classification algorithm:

- Given a query instance x_q to be classified,
  - Let $x_1 \ldots x_k$ denote the k instances from *training_examples* that are nearest to x_q
  - Return

$f(x_i)$ → actual class label of i^{th} neighbor

new output

$$\hat{f}(x_q) \leftarrow \underset{v \in V}{\operatorname{argmax}} \sum_{i=1}^{k} \delta(v, f(x_i))$$

where $\delta(a, b) = 1$ if $a = b$ and where $\delta(a, b) = 0$ otherwise.

Approximate a real-valued target function $f : \Re^n \rightarrow \Re$

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} f(x_i)}{k}$$

New output

Summary

- **Training**: Just store the data.
- **Prediction**:
  - Find the k nearest neighbors to the test instance.
  - Count how many times each class label appears among them.
  - Predict the class with the **majority vote**.

📌 Example

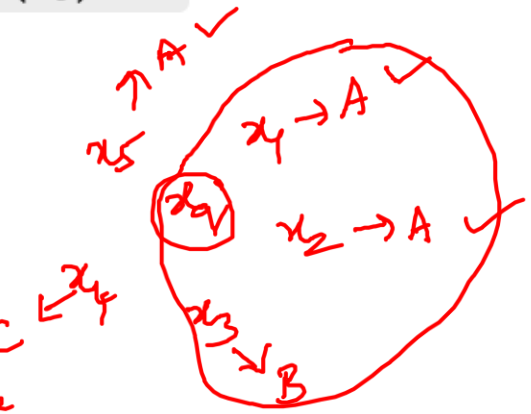
Let's say:

- You have 3 classes: {A, B, C} → so $V = \{A, B, C\}$
- For a test point x_q, its 5 nearest neighbors have labels:

`f(x₁) = A`, `f(x₂) = A`, `f(x₃) = B`, `f(x₄) = C`, `f(x₅) = A`

Then:

- Count of A: 3
- Count of B: 1
- Count of C: 1



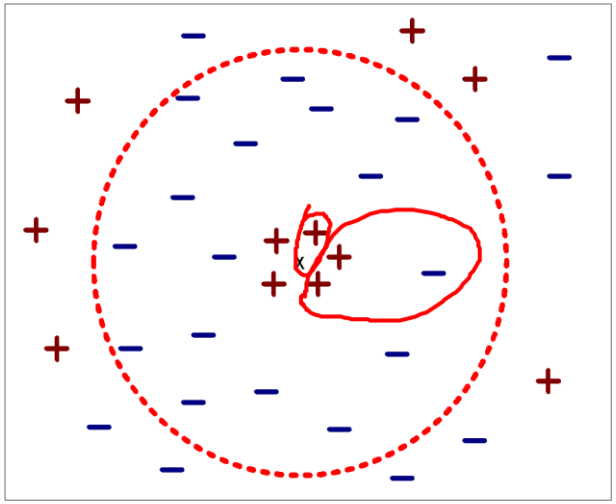
$$\hat{f}(x_q) = \arg \max_{v \in \{A,B,C\}} \sum_{i=1}^{5} \delta(v, f(x_i)) = A$$

Instance Based Learning

K-Nearest Neighbor : Hyper parameter "K"

Choosing the value of k:

Rule of thumb:
K = sqrt(N)
N: number of training points



100 - data points

$k = \sqrt{100} = 10$

$k = \sqrt{30}$

$k = 10$

optimum value.

x_1 1.33

x_2 2.71

$d(x_1, x_2) = |1.33 - 2.71|$

$\sqrt{(x_1 - x_2)^2}$

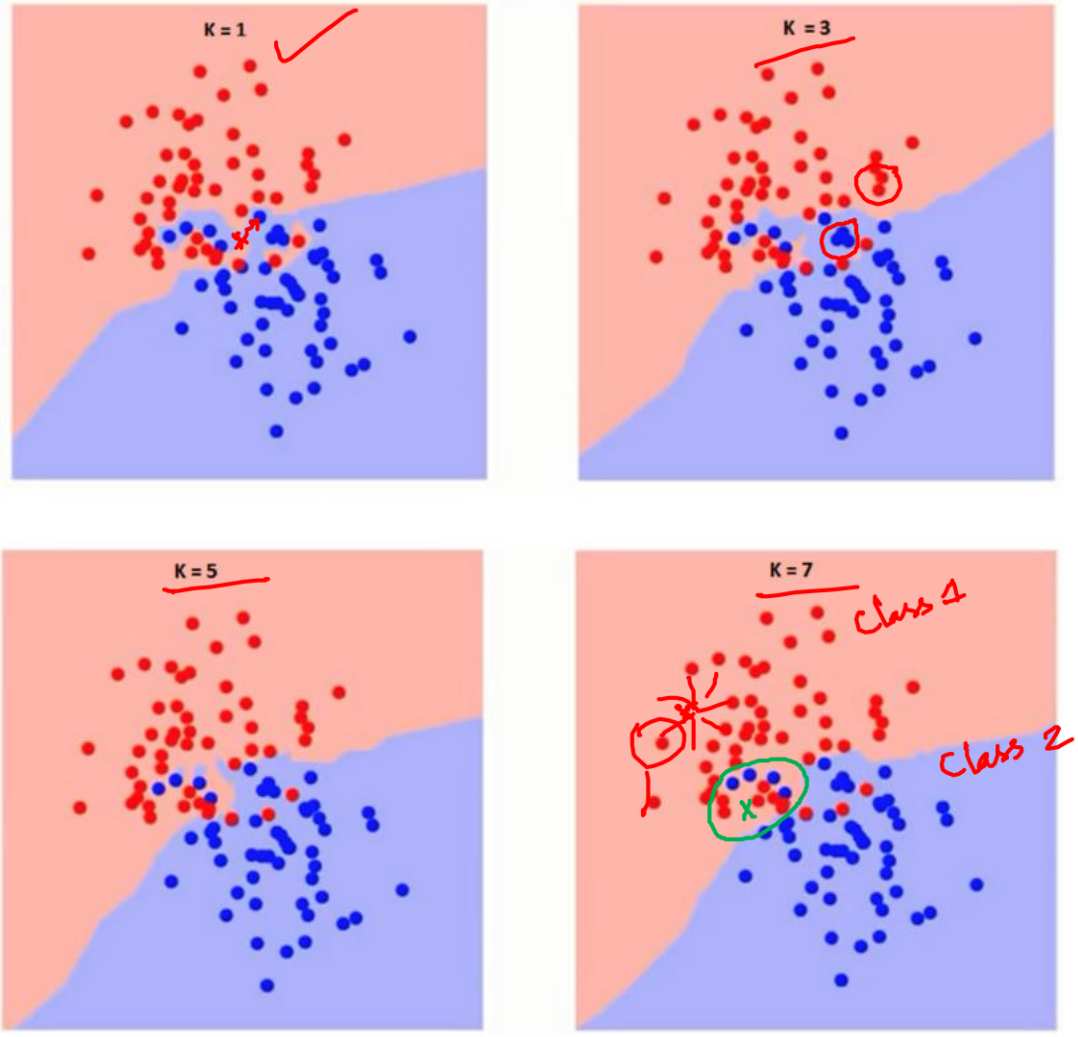
Instance Based Learning

innovate achieve lead

K-Nearest Neighbor : Hyper parameter "K" : Elbow Method

- Compute sum of squares error (SSE) or any other error function for varying values of K (1 to a reasonable X) and plot against K
- In the plot, the elbow (see pic) gives the value of K beyond which the error function plot almost flattens
- As K approaches the total number of instances in the set, error function drops down to '0'

Act pred $(x - x_i)^2$

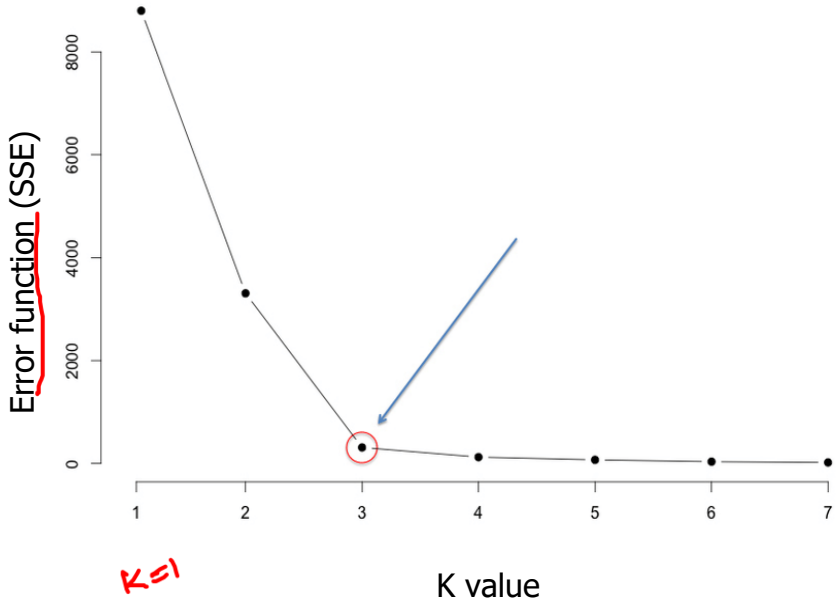


Instance Based Learning

innovate achieve lead

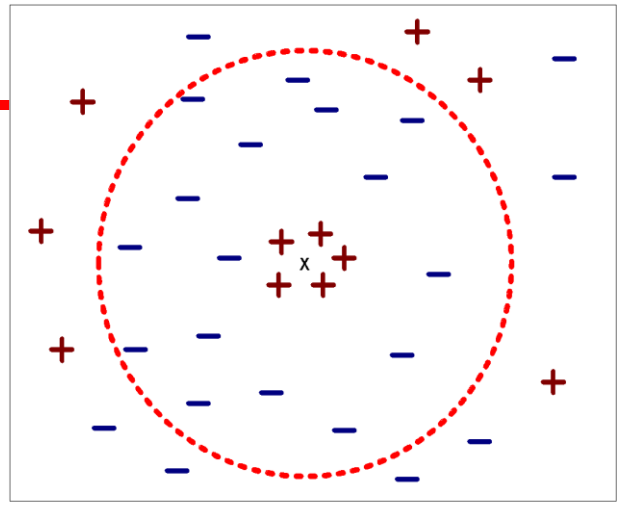
K-Nearest Neighbor : Hyper parameter "K" : Elbow Method

- Compute sum of squares error (SSE) or any other error function for varying values of K (1 to a reasonable X) and plot against K
- In the plot, the elbow (see pic) gives the value of K beyond which the error function plot almost flattens
- As K approaches the total number of instances in the set, error function drops down to '0'



Instance Based Learning

K-Nearest Neighbor : Variation



- Distance Weighted K-NN algorithm

contribution of each of the k nearest neighbors is weighted accorded to their distance to x_q

- **discrete-valued target functions**

$$\hat{f}(x_q) \leftarrow \underset{v \in V}{argmax} \sum_{i=1}^{k} w_i \delta(v, f(x_i))$$

where $w_i \equiv \frac{1}{d(x_q, x_i)^2}$ and $\hat{f}(x_q) = f(x_i)$ if $x_q = x_i$

- **continuous-valued target function:**

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} w_i f(x_i)}{\sum_{i=1}^{k} w_i}$$

Kernel Functions

$w_i = K(d(x_q, x_i))$

$K(d(x_q, x_i)) = 1/\ d(x_q, x_i)^2$

$K(d(x_q, x_i)) = 1/(d_0 + d(x_q, x_i))^2$

$K(d(x_q, x_i)) = \exp(-(d(x_q, x_i)/\sigma_0)^2)$

$d(x_q, x_i)$ is the distance between x_q and x_i

Problem Type – 1

Consider the following training set in 2-dimensional Euclidean space. What is the class of the point (1, 1) if 7NN classifier is considered? If the value of K is reduced whether the class will change? (Consider K=3 and K=5). What should be the final class if the above 3 values of K are considered?

Point	Coordinate	Class
X1	(-1, 1)	Negative
X2	(0, 1)	Positive
X3	(0, 2)	Negative
X4	(1, -1)	Negative
X5	(1, 0)	Positive
X6	(1, 2)	Positive
X7	(2, 2)	Negative
X8	(2, 3)	Positive

Distance from (1,1)
2
1
1.414
2
1
1
1.41
2.236

Point	Distance	Class
X2	1	Positive
X5	1	Positive
X6	1	Positive
X7	1.41	Negative
X3	1.414	Negative
X1	2	Negative
X4	2	Negative
X8	2.236	Positive

$d[x_{new}, x_1] = d[(1,1) \& (-1,1)]$ (x_1, y_1 and x_2, y_2)

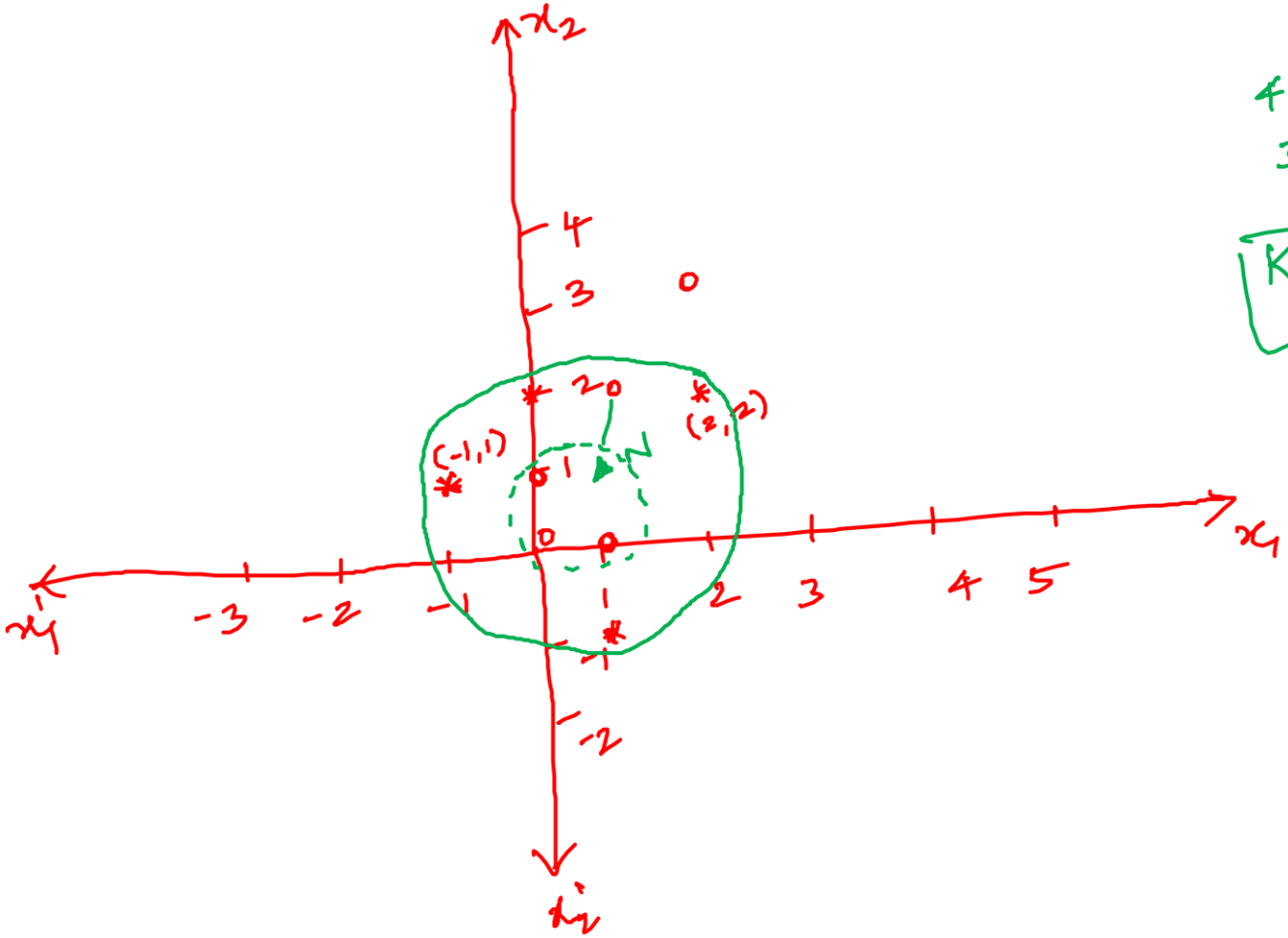
K=8

$\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$

when $y = 0$ ↓ x-axis

N → *

P → o



4 → N

3 → P

K = T

$d(i,j) = \sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2}$

Point	Coordinate	Class	Distance from (1, 1)	
X1	(−1,1)	Negative	**2**	$\sqrt{(-1-1)^2 + (1-1)^2} = \sqrt{(-2)^2 + 0^2} = \sqrt{4} = 2$
X2	(0,1)	Positive	**1**	$\sqrt{(0-1)^2 + (1-1)^2} = \sqrt{(-1)^2 + 0^2} = \sqrt{1} = 1$
X3	(0,2)	Negative	**1.414**	$\sqrt{(0-1)^2 + (2-1)^2} = \sqrt{(-1)^2 + 1^2} = \sqrt{2} \approx 1.414$
X4	(1,−1)	Negative	**2**	$\sqrt{(1-1)^2 + (-1-1)^2} = \sqrt{0^2 + (-2)^2} = \sqrt{4} = 2$
X5	(1,0)	Positive	**1**	$\sqrt{(1-1)^2 + (0-1)^2} = \sqrt{0^2 + (-1)^2} = \sqrt{1} = 1$
X6	(1,2)	Positive	**1**	$\sqrt{(1-1)^2 + (2-1)^2} = \sqrt{0^2 + 1^2} = \sqrt{1} = 1$
X7	(2,2)	Negative	**1.41**	$\sqrt{(2-1)^2 + (2-1)^2} = \sqrt{1^2 + 1^2} = \sqrt{2} \approx 1.414$
X8	(2,3)	Positive	**2.236**	$\sqrt{(2-1)^2 + (3-1)^2} = \sqrt{1^2 + 2^2} = \sqrt{5} \approx 2.236$

KNN Classification (Determining the Class of (1, 1))

The final step in KNN is to select a value for K (the number of nearest neighbors to consider) and then assign the class of the majority of those K neighbors to the unknown point (1, 1). Common choices for K are odd numbers to avoid ties.

Case 1: K=1

Nearest Neighbor: X2, X5, and X6 all have the minimum distance of 1.

Class: The point (1, 1) would be classified as **Positive** (since X2, X5, X6 are all Positive). In a real-world scenario with a distance tie, a tie-breaking rule (like randomly selecting one or taking the average) might be used, but since all tied neighbors share the same class, the result is clear.

Case 2: K=3

3 Nearest Neighbors: X2, X5, X6 (all distance 1).

Classes: 3 Positive, 0 Negative.

Majority Class: Positive.

Case 3: K=5

5 Nearest Neighbors: X2, X5, X6, X7, X3.

Classes: 3 Positive (X2, X5, X6) and 2 Negative (X7, X3).

Majority Class: Positive.

Based on the provided data, for any reasonable small value of K (e.g., K=1,3,5), the unknown point **(1, 1)** would be classified as **Positive** due to the concentration of positive points at the shortest distance.

Problem Type – 2

Based on the information given in the table below, find the customer type of x_q=C4 using K-NN. In given example consider all the instances and use locally weighted K-NN algorithm with kernel $K(d(x_q,x_i)) = 1/ d(x_q,x_i)^2$.

Apply min-max normalization on income to obtain [0,1] range. Consider profession and Region as nominal. Consider :Locality as ordinal variable with ranking order of [Village, Small Town, Suburban, Metropolitan]. Give equal weight to each attribute.

Customer	Income	Profession	Region	Locality	Category
C0	60000	Doctor	Hindi	Village	**L1**
C1	70000	Doctor	Bengali	Village	**L2**
C2	60000	Carpenter	Hindi	Suburban	**L2**
C3	80000	Doctor	Bhojpuri	Metropolitan	**L2**
C5	80000	Data Scientist	Hindi	Small Town	**L1**
C4	**50000**	**Data Scientist**	**Hindi**	**Small Town**	??

Problem Type – 2

Based on the information given in the table below, find the customer type of x_q=C4 using K-NN. In given example consider all the instances and use locally weighted K-NN algorithm with kernel $K(d(x_q,x_i)) = 1/ d(x_q,x_i)^2$.

Apply min-max normalization on income to obtain [0,1] range. Consider profession and Region as nominal. Consider :Locality as ordinal variable with ranking order of [Village, Small Town, Suburban, Metropolitan]. Give equal weight to each attribute.

Customer	Income	Profession	Region	Scaled Locality	Category	Distance	Weight
C0	0.33	Doctor	Hindi	1 □ 0	L1	0.97	1.06
C1	0.67	Doctor	Bengali	1 □ 0	L2	1.75	0.33
C2	0.33	Carpenter	Hindi	3 □ 0.67	L2	0.97	1.06
C3	1	Doctor	Bhojpuri	4 □ 1	L2	2.2	0.21
C5	1	Data Scientist	Hindi	2 □ 0.33	L1	1	1
C4	0	Data Scientist	Hindi	2 □ 0.33		0 (NA)	

d(C4, C2)

= distance w.r.t to numerical attributes + distance w.r.t Nominal attributes

= Euclidean distance on (Income, Scaled Locality) + Categorical distance on attributes (Profession, Region)

= $\sqrt{((Income_{c4}-Income_{c2})^{"2"} +(Locality_{c4}-Locaility_{c2})^2}$)+(No.of.Categorical features -No.of.Matches between C4 & C2)/(No.of.Categorical Features)

= $\sqrt{((0-0.33)^{"2"} +(0.33-0.67)^2}$)+(2 -1)/2

= 0.47 + 0.5 = 0.97

Problem Type – 2

Based on the information given in the table below, find the customer type of x_q=C4 using K-NN. In given example consider all the instances and use locally weighted K-NN algorithm with kernel $K(d(x_q,x_i)) = 1/ d(x_q,x_i)^2$.

Apply min-max normalization on income to obtain [0,1] range. Consider profession and Region as nominal. Consider :Locality as ordinal variable with ranking order of [Village, Small Town, Suburban, Metropolitan]. Give equal weight to each attribute.

Customer	Income	Profession	Region	Scaled Locality	Category	Distance	Weight
C0	0.33	Doctor	Hindi	1 □ 0	L1	0.97	1.06 ✓
C1	0.67	Doctor	Bengali	1 □ 0	L2 ✓	1.75 ✓	0.33
C2	0.33	Carpenter	Hindi	3 □ 0.67	L2 ✓	0.97	1.06
C3	1	Doctor	Bhojpuri	4 □ 1	L2 ✓	2.2	0.21
C5	1	Data Scientist	Hindi	2 □ 0.33	L1	1	1
C4	0	Data Scientist	Hindi	2 □ 0.33		0 (NA)	

$w_i = K[d(x_q,x_i)] = \frac{1}{[d(x_q,x_i)]^2}$

$w_1 = \frac{1}{(0.97)^2} = 1.06$

$w_2 = \frac{1}{(1.75)^2} = 0.33$

Point	Kernel / Weight	Class
C5	1	L1
C0	1.06	L1
C2	1.06	L2
C1	0.33	L2
C3	**0.21**	L2

Inference : Without weighted KNN, L2 is the majority voted class.

But with weighted distances, L1 class instances are more similar to C4

Sum weights by category

L1: $1.06 + 1.000 = 2.06$

L2: 1.6

Conclusion:

Weighted vote for L1: 2.06

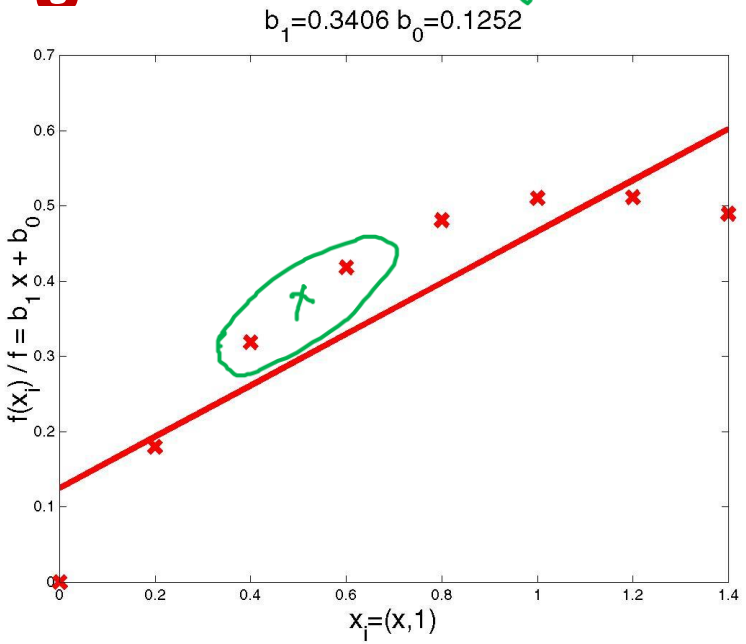
Weighted vote for L2: 1.6

Since **L1** has the higher weighted vote, the **weighted K-NN classification for C4 is Category L1.**

$ax + b = c$

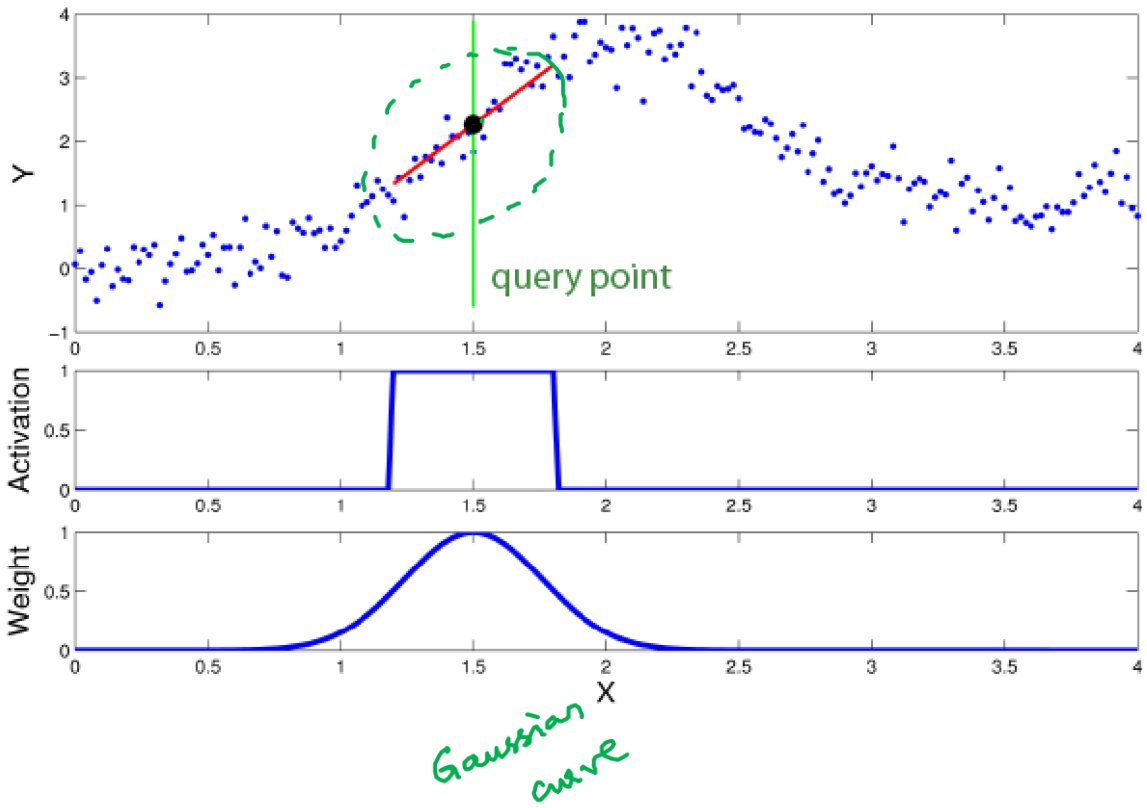
Locally Weighted Regression

predicting values



Locally Weighted Regression

- Locally – Function approximated based on data near query point
- Weighted – Contribution by each training example is weighted by its distance from query point
- Regression- Approximates real-valued target function
- ***Residual*** is the error in approximating the target function. $\hat{f}(x) - f(x)$
- ***Kernel function*** is the function of distance that is used to determine the weight of each training example. In other words, the kernel function is the function ***K*** such that $w_i = K(d(x_i, x_q))$



Source Credit : https://www.ias.informatik.tu-darmstadt.de/uploads/Teaching/AutonomousLearningSystems/Englert_ALS_2012.pdf

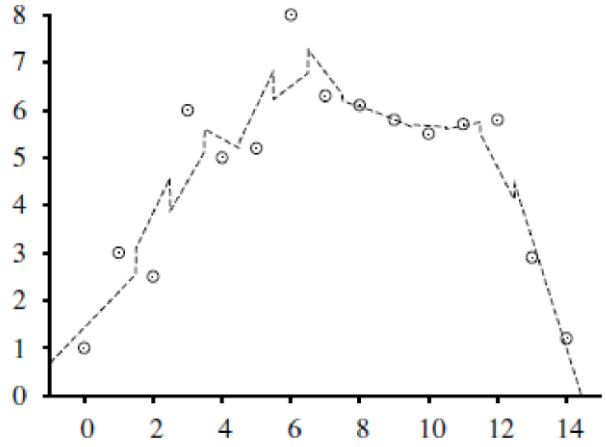
Locally Weighted Regression

- The nearest-neighbor approaches can be thought of as approximating the target function f ***(x)*** at the single query point ***x*** = x_q

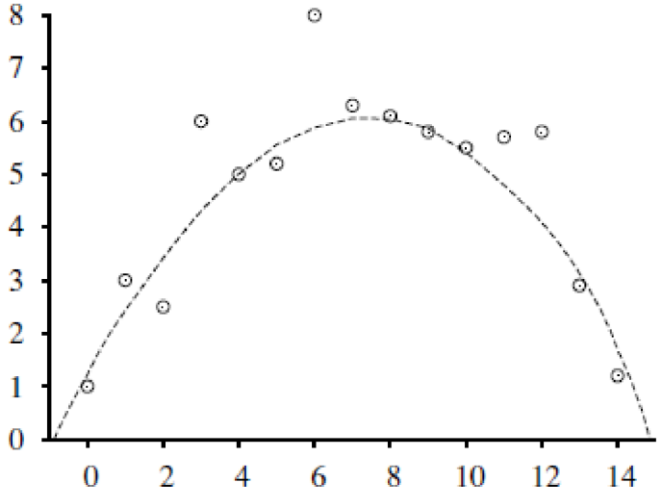
- Locally weighted regression is a generalization of this approach. It constructs an explicit approximation to f over a local region surrounding x_q

- Approximate the target function in the neighborhood surrounding x, using a linear function, a quadratic function, a multilayer neural network, or some other functional form.

3-nearest-neighbors linear regression



Locally weighted regression with a quadratic kernel



Locally weighted linear regression

- target function is approximated using a **linear function**

$$\hat{f}(x) = w_0 + w_1 a_1(x) + ... + w_n a_n(x)$$

$y = a + b_1 x_1 + b_2 x_2$

- methods like **gradient descent** can be used to calculate the coefficients $w_0, w_1, ..., w_n$ to minimize the error in fitting such linear functions
- ANNs require a global approximation to the target function
- here, just a local approximation is needed

$\Rightarrow$ the error function has to be redefined

Locally weighted linear regression

- possibilities to redefine the error criterion E
  1. Minimize the squared error over just the k nearest neighbors

$$E_1(x_q) \equiv \frac{1}{2} \sum_{x \in k \text{ nearest neigbors}} (f(x) - \hat{f}(x))^2$$

  2. Minimize the squared error over the entire set D, while weighting the error of each training example by some decreasing function K of its distance from x_q

$$E_2(x_q) \equiv \frac{1}{2} \sum_{x \in D} (f(x) - \hat{f}(x))^2 \cdot K(d(x_q, x))$$

  3. Combine 1 and 2

$$E_3(x_q) \equiv \frac{1}{2} \sum_{x \in k \text{ nearest neighbors}} (f(x) - \hat{f}(x))^2 \cdot K(d(x_q, x))$$

Locally weighted linear regression

- For a given query point $\mathbf{x}_q$ we solve the following weighted regression problem using gradient descent training rule:

$$\Delta w_j = \eta \sum_{x \in k \text{ nearest nbrs of } x_q} K(d(x_q, x)) \; (f(x) - \hat{f}(x)) \; a_j(x)$$

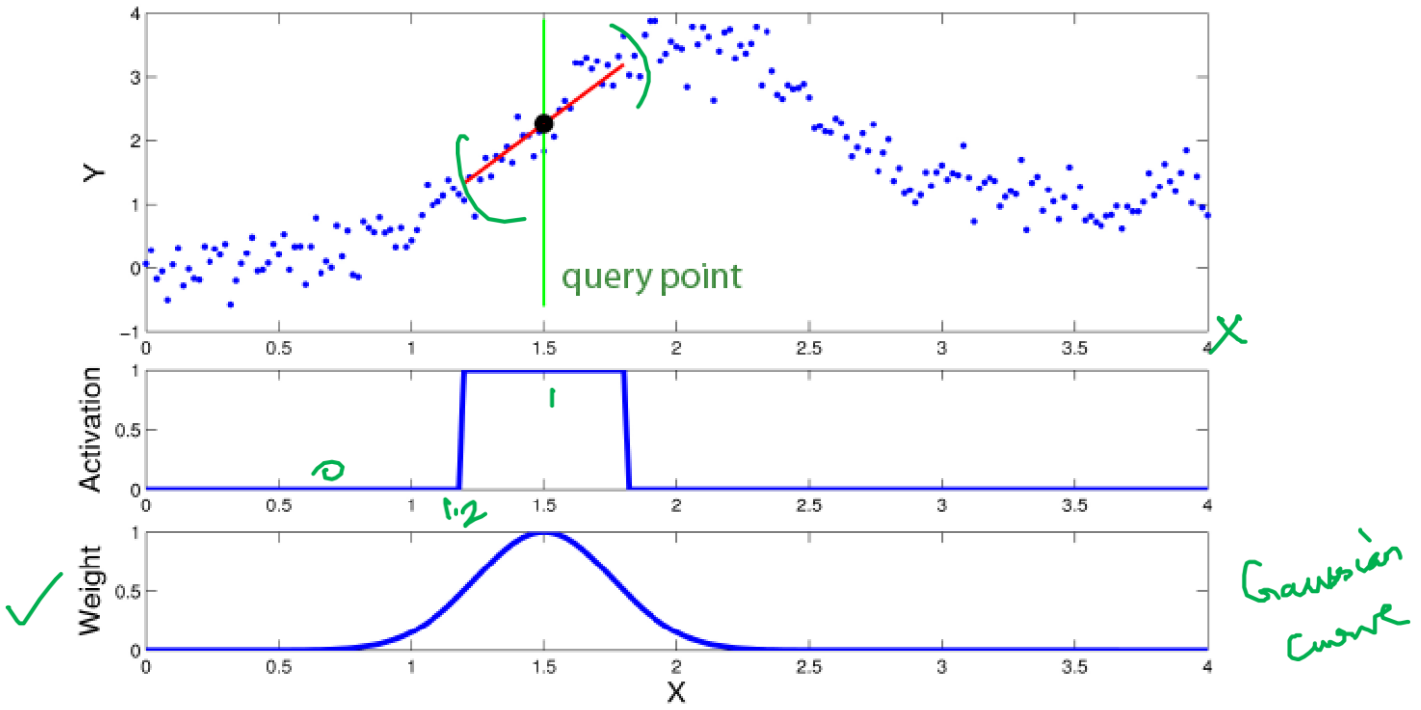
- Note that we need to solve a new regression problem for *every* query point—that's what it means to be *local*.
- In ordinary linear regression, we solved the regression problem once, globally, and then used the same $h_{\mathbf{w}}$ for any query point.

$$w_i \leftarrow w_i + \Delta w_i$$

Note: + sign in weight update rule as in Δw_i (actual-prediction) is used.

If we use (prediction – actual) then instead of '+', '-' sign will be used.

Example



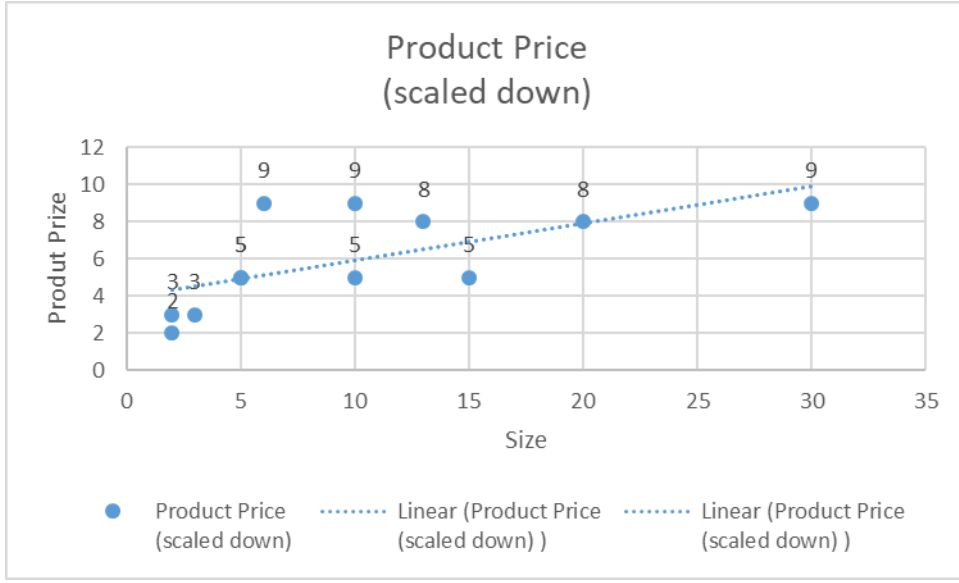
- The upper graphic the set of data points (x,y) (blue dots), query point (green line), local linear model (red line) and prediction (black dot).
- The graphic in the middle shows the activation area of the model.
- The corresponding weighting kernel (receptive field) is shown in the bottom graphic.

Source Credit : https://www.ias.informatik.tu-darmstadt.de/uploads/Teaching/AutonomousLearningSystems/Englert_ALS_2012.pdf

Problem Type - 3

Consider the following training set in 2-dimensional Euclidean space. What is the prize of the product with size 7 if 5-NN is considered? Use the kernel of the form $K(d(x_q,x_i)) = \exp(- d(x_q,x_i)^2 / 2b^2)$ where b=1. Use the linear regression of the form y = 3.5 + 0.8 x and the learning rate of 0.2 to apply the gradient descent and update the weights at the end of first iteration

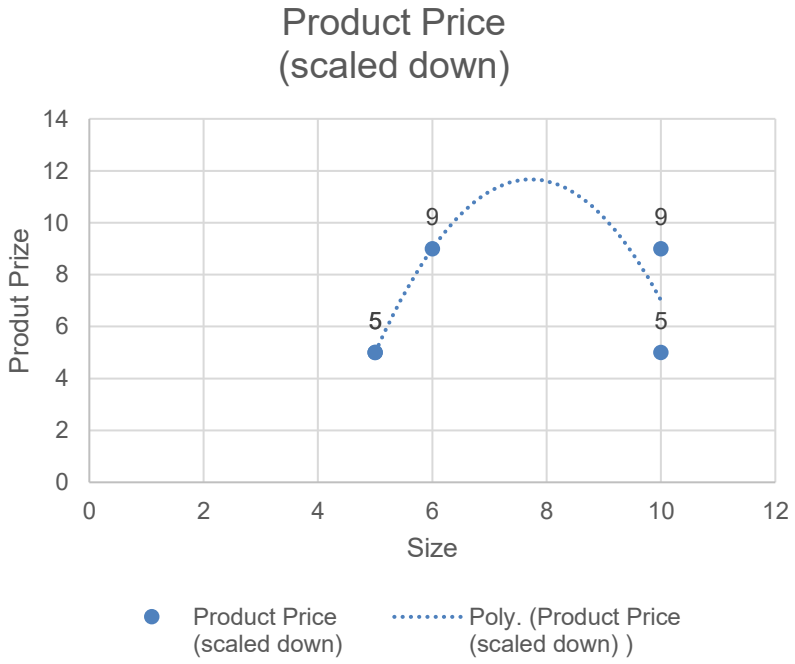
Size	Product Price (scaled down)
10	5
5	5
2	2
3	3
5	5
15	5
30	9
20	8
6	9
2	3
10	9
13	8



Problem Type - 3

Consider the following training set in 2-dimensional Euclidean space. What is the prize of the product with size 7 if 5-NN is considered? Use the kernel of the form $K(d(x_q,x_i)) = \exp(- d(x_q,x_i)^2 / 2b^2)$ where b=1. Use the linear regression of the form $y = 3.5 + 0.8\ x$ and the learning rate of 0.2 to apply the gradient descent and update the weights at the end of first iteration

Size	Product Price (scaled down)
10	5
5	5
2	2
3	3
5	5
15	5
30	9
20	8
6	9
2	3
10	9
13	8



Problem Type - 3

Use either of below formula :

$$W_{j-new} = W_{j-old} + \alpha * \sum_{i=1}^{K\ Neighbors} K(d(x_q,x_i))(\ f(x_i) - f\hat{}(x_i)\ )a_j(x_i)$$

$$W_{j-new} = W_{j-old} - \alpha * \sum_{i=1}^{K\ Neighbors} K(d(xq,xi))(f\hat{}(x_i) - f(x_i))a_j(x_i)$$

Size	Product Price (scaled down)	d(xq, xi)	$K(d(x_q,x_i))$
10	5	3	0.01
5	5	2	0.14
2	2	5	0
3	3	4	0
5	5	2	0.14
15	5	8	0
30	9	23	0
20	8	13	0
6	9	1	0.61
2	3	5	0
10	9	3	0.01
13	8	6	0

Apply gradient descent where the gradient is weighed by the kernel output.
Answer :
Updated weights after 1st iteration
w0 = 3.43, w1= 0.43

Price Prediction	
6.459515	After 1st iteration of GD
9.1	Before GD

When to Consider Nearest Neighbors

- Suitable for Low dimensional datasets
- Local information can yield highly adaptive behavior
- Lots of training data (distance-weighted KNN)
    - Training is very fast
- Learn complex target functions
    - Do not lose information
- Noisy training data (distance-weighted KNN)
    - by taking the weighted average of the k neighbors nearest to the query point, it can smooth out the impact of isolated noisy training examples.

Nearest-Neighbor Classifiers: Challenges

- The value of *k*, the number of nearest neighbors to retrieve
- Choice of Distance Metric to compute distance between records
- Expensive, Slow at query time
  - To determine the nearest neighbour of a query point *q*, must compute the distance to all *N* training examples
    - + Pre-sort training examples into fast data structures (kd-trees)
    - + Remove redundant data (condensing)
- Storage Requirements
  - Must store all training data **P**
    - + Remove redundant data (condensing)

Additional Exercise for Students

Size	Rating (1-10)	Product Price (scaled down)
10	5	5
5	2	5
2	7	2
3	6	3
5	4	5
15	8	5
30	4	9
20	8	8
6	10	9
2	2	3
10	4	9
13	6	8

1. Consider the following training set in 2-dimensional Euclidean space. What is the prize of the product with size 12 if 7-NN is considered? Use the kernel of the form $K(d(x_q,x_i)) = 1/ d(x_q,x_i)^2$. Use the linear regression of the form $y = 2 + 1.5\ x$ and the learning rate of 0.6 to apply the gradient descent and update the weights at the end of first iteration. Use Manhattan Distance for the distance calculation.
2. Try to scale all the features with min-max of [0-5] and redo the modelling under previous question. Observe to Interpret the influence of the scaling
3. Find the MSE of the K-NNs from the results of both above parts separately & state your observations
4. If the Product with features (13,6) and (30,4) are outliers then interpret the effect of the same in the K-NN modelling, before and after removing them with numerical justification.

Lab : Practise :

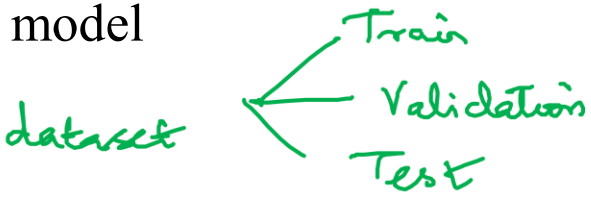
Take any BITS virtual Lab dataset and experiment on below cases

- **Tune KNN**. Try different values of *k* to see if you can improve the performance of the algorithm.
- **Use weighted KNN** and check the performance
- **Regression**. Apply it to a regression predictive modelling problem (e.g. predict a numerical value)
- **More Distance Measures**. Implement other distance measures such as Hamming distance, Manhattan distance and Minkowski distance.
- **Data Preparation**. Distance measures are strongly affected by the scale of the input data. Experiment with and without standardization and other data preparation methods in order to improve results.

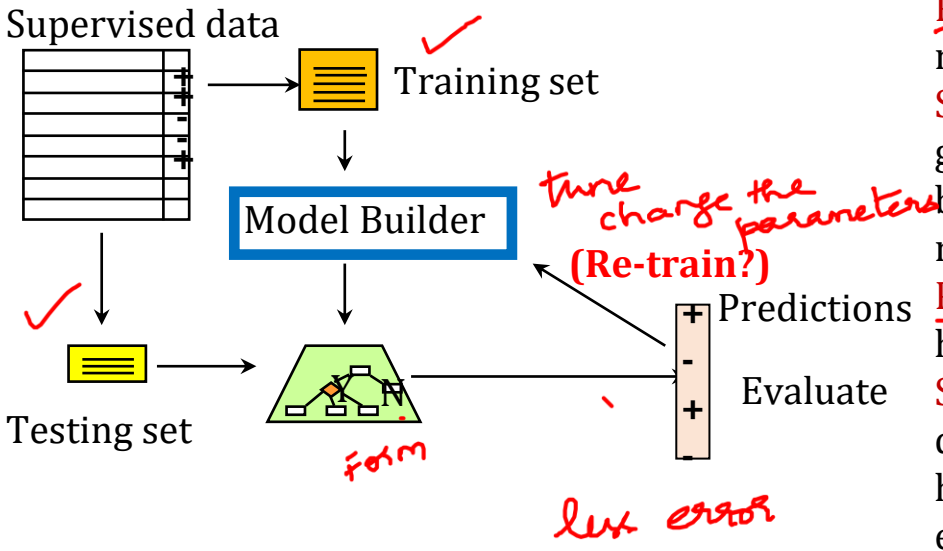
Evaluating Classifier Performance

Model selection and assessment

- Model selection: estimating the performance of different models in order to choose the best one.
- Model assessment: having chosen a final model, estimating its prediction error (generalization error) on new data.
- To estimate generalization error (measure the quality of inductive bias), we need data unseen during training. We split the data as
  - Training set : used to fit the models
  - Validation set : used to estimate prediction error for model selection, tune hyperparameters, feature selection
  - Test (publication) set : is used for assessment of the generalization error of the final chosen model



Hyper parameter tuning using test set



Problem 1: Hesitating between two models (e.g linear model and a polynomial model) ?
Solution 1: train both and compare how well they generalize using the test set: linear model generalizes better but looks like overfitting So, apply some regularization
Problem 2: how to choose the value of regularization hyper parameter?
Solution 2: train 100 different models using 100 different values for this hyper parameter. Choose best hyper parameter value with the lowest generalization error (5%). Great!!!: launch into production

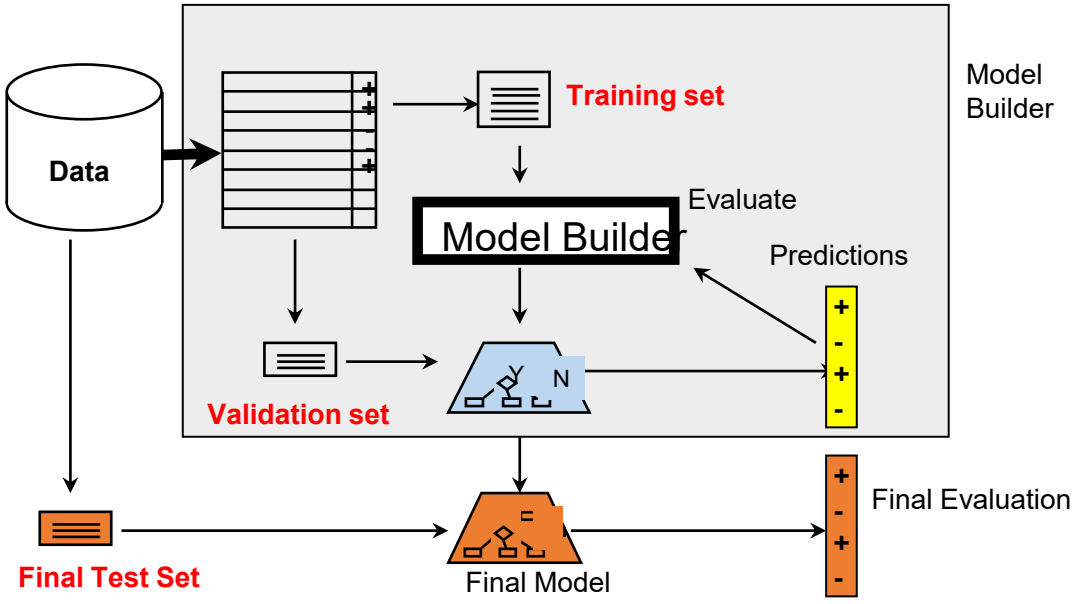
Problem 3: Error now is increased to 5% errors. Why?
Answer 3: Generalization error was measured multiple times on the test set, So model and hyper parameters are adapted to produce the best model for that particular set, unlikely to perform as well on new data.

Learning : The test data can't be used for parameter tuning!

Train, Validation, Test split

Holdout validation (holdout or development,* or *dev set)

1. Train multiple models with various hyper parameters on the reduced training set (i.e., the full training set minus the validation set)
2. Select the model that performs best on the validation set
3. After this holdout validation process, you train the best model on the full training set (including the validation set), to get the final model.
4. Lastly, evaluate this final model on the test set to get an estimate of the generalization error



Example:

A dataset of 10000 houses

Step 1: Split the data into 3 parts:

80:20

Training data set
- Training sets (70%) → 700 houses
- Validation set (15%) → 150 houses
- Test set (15%) → 150 houses

Step 2: Train the multiple model

Model A: Linear Regression

Model B: Polynomial reg

Model C: Decision Trees

10 students
- 5 → good ML
- 3 → Avg ML
- 2 → no ML

Step 3: Use validation data set to choose best model

Model A → error = 25000

Model B → error = 20,000 ✓

Model C → error = 22,000

Step 4: Retrain the final model

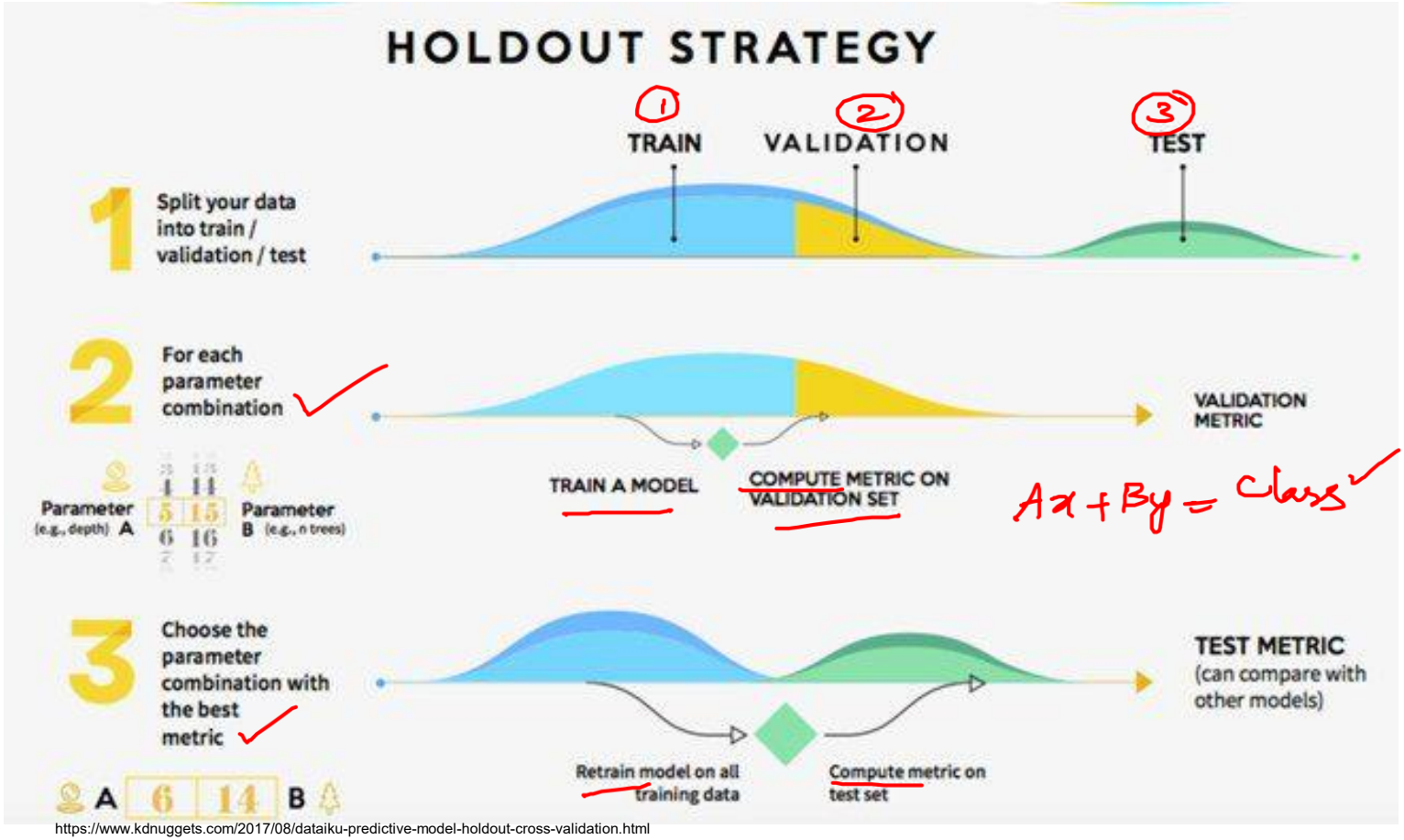
using Training + Validation data set

O/p → Best parameters.

Step 5: Evaluate the model using test set

Test error = 21,000

Train-Test-Validation



https://www.kdnuggets.com/2017/08/dataiku-predictive-model-holdout-cross-validation.html

Summary:

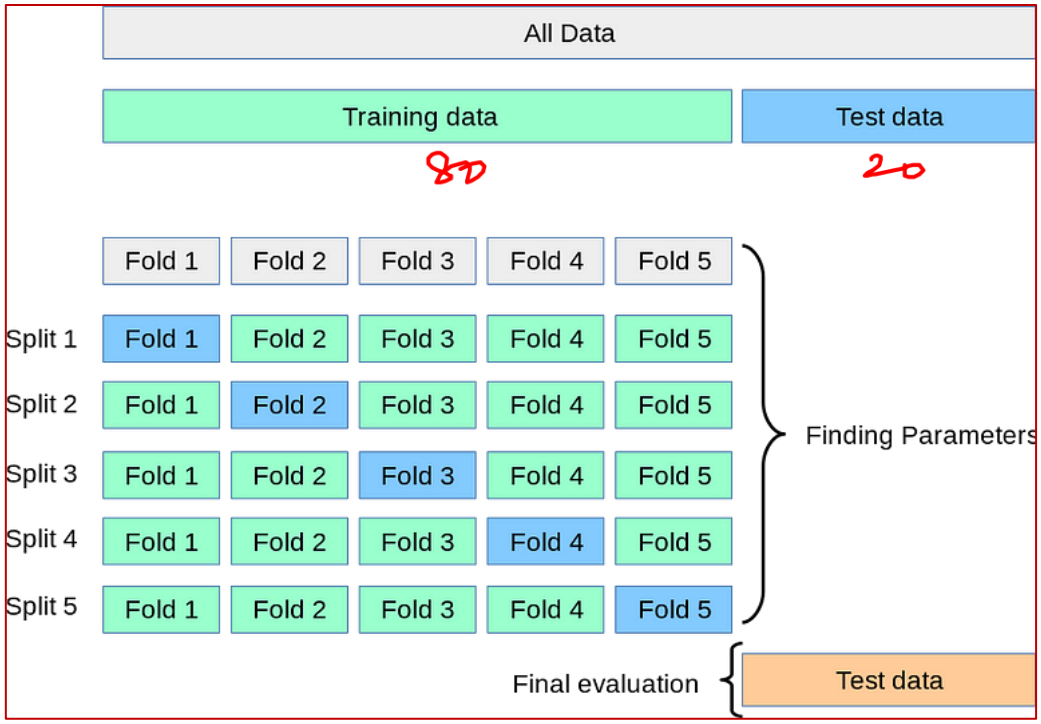
- **Training set** → learns patterns
- **Validation set** → helps choose the best model
- **Test set** → gives a fair final score → Error

$2x+3y=1$ → find model

Linear regression
a, b, c
finalizing the values of a, b, c

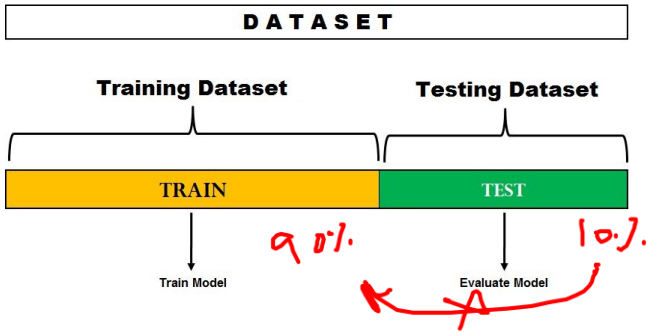
Making most out of the data

- We don't want to touch the test set until we are ready to launch a model we are confident about
- If the validation set is too small, then model evaluations will be imprecise
  - Larger the test data the more accurate the error estimate
- If the validation set is too large, then the remaining training set will be much smaller than the full training set.
  - Larger the training data the better the classifier
- **Solution: cross validation**

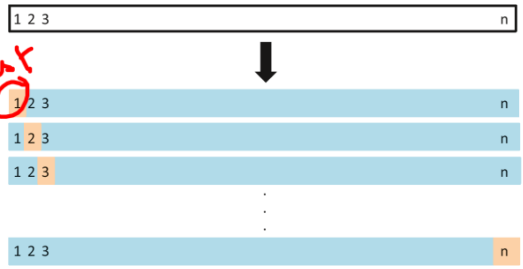


Cross-Validation techniques

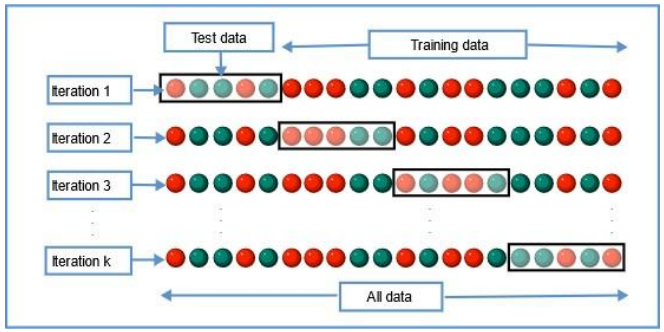
Hold Out method



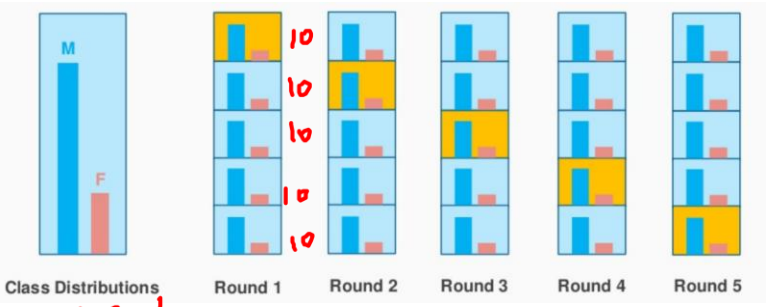
Leave One Out Cross-Validation



K-Fold Cross-Validation

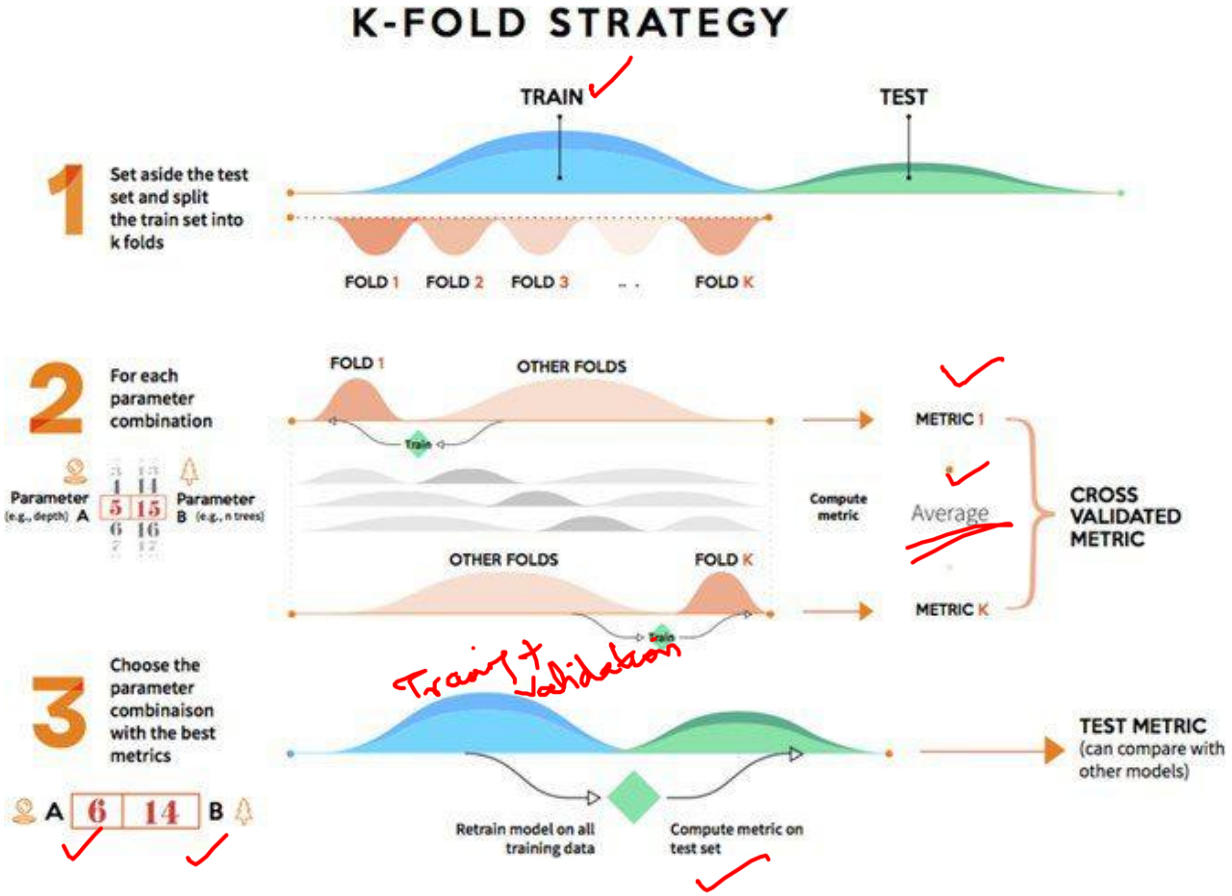


Stratified K-Fold Cross-Validation



https://www.analyticsvidhya.com/blog/2021/05/4-ways-to-evaluate-your-machine-learning-model-cross-validation-techniques-with-python-code/

Cross-Validation for hyper parameter tuning



https://www.kdnuggets.com/2017/08/dataiku-predictive-model-holdout-cross-validation.html

Estimating generalization error : cross validation set Vs test set

- Training set performance is not a good indicator of generalization error
  - A more complex model overfits, a less complex one underfits
  - Which model do I select?
- Test set
  - Typically 80%, 20%
  - Wastes valuable labeled data
- Cross validation
  - In Sklearn : cross_val_score() (default is accuracy)
  - cross_val_score will automatically split data into train and test
  - fit the model on train data
  - score on test data
  - For 10 folds, 10 scores will be returned in scores variable. Then take average of that

```
model=DecisionTreeClassifier()
scores = cross_val_score(model, X, y, cv=10)
avg_score = np.mean(score_array)
print(avg_score)
```

Thank you !

Required Reading for completed session :

T1 - Chapter # 8 (Tom M. Mitchell, Machine Learning)

Next Session Plan :

Module 7 : Support vector machines

Since the math basics will be covered in detail in math course, ML course is intended to discuss the application & interpretation of the model

Lagrange Multipliers
KKT – Conditions
Primal – Dual for SVM
Kernel for Non-Linear SVM